EUROPEAN PATENT APPLICATION

(54) **Method and apparatus for protecting a computer system from computer viruses.**

(57) The present invention is an apparatus for protection against attack by computer virus is effectuated by detecting virus and preventing virus attack at boot time. This is achieved by disabling modifications to storage devices of the system before the booting process and by detecting the presence of virus by checking integrity of the interrupt vectors of the system.
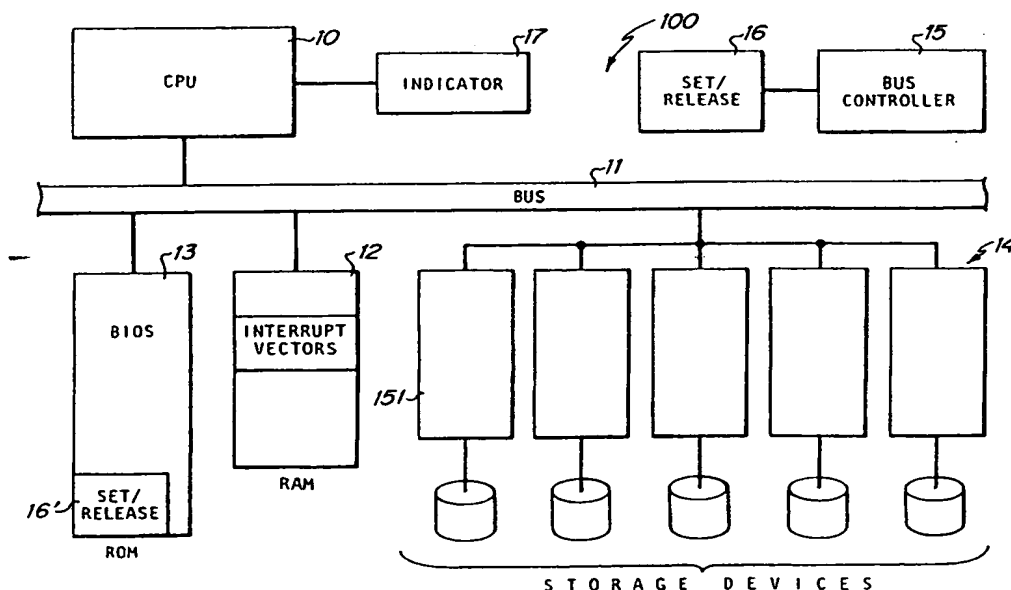
EP 0 510 244 A1

Figure 1

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method and apparatus for protecting a computer system against vandalism commonly known as computer viruses.

2. Description of the Related Art

As the world becomes more and more computerized, one of the great concerns of the data processing industry has been possible attacks on a computer system by viruses. In less serious cases, an attack by a computer virus may force an organization to replicate or recover data and files before it can resume normal operation. In more serious cases, data and files destroyed by a virus may become unrecoverable, forever shutting down the organization's operation. In the most serious cases, the integrity of an organization's data bases may have been attacked without warning, and the organization continues operation using inaccurate data, resulting in injuries, losses and damages.

The virus problem is aggravated because typical computer viruses are, like biological viruses, capable of self-replication, and can be infected through connection to networks. Viruses can also attach themselves to application and/or system programs. Moreover, a relatively harmless virus may undergo "mutation" (modified during its residence in a computer system) to become a fatal one.

One common way which viruses attack a computer system is by changing the interrupt vectors in the system so that the virus can get its hands on the operation of the system and the I/O devices. Therefore, one prior art anti-virus method sets hooks on interrupt vectors and interrupt program instructions in order to detect attempts by a virus to change the interrupt vectors.

In another prior art anti-virus method, programs to be executed in the system first undergo a self-test to detect whether the program has been modified. If a program has been modified, it may have been infected and execution of the program in the system is inhibited.

In yet another prior art anti-virus method, predefined characteristics (e.g., check sum) of each program in its uninfected state are stored. Before a program is executed, these characteristics are regenerated and compared with their corresponding stored counterparts to determine whether it has been modified.

However, these prior art methods are unsatisfactory for three reasons. First, the operation of these prior art method requires assistance of the computer's operating system (e.g., DOS). Since the operating system can only function after successfully booting the computer system, these prior art methods cannot protect the computer from viruses that attack when a computer system boots. i.e., before installation of the operating system. Second, the prior art is unable to prevent viruses which bypass the operating system and operate directly on the hardware or the BIOS. Third, existing anti-virus systems cannot prevent viruses which reside in the operating system modules (e.g., IO.SYS, MSDOS.SYS, and COMMAND.COM).

Therefore, there is a need for a method and apparatus for protecting computer systems against viruses, not only by preventing viruses from entering into a computer system, but also by removing and disabling a virus that already resides in the computer system. Also, there is a need for protection from viruses during the period when the systems boots up.

SUMMARY OF THE INVENTION

The preferred embodiment of the present invention for protecting a computer system from viruses comprises means for inhibiting before booting of the system, storage of data into storage devices of the computer system, means for checking integrity of programs already loaded into the system, and means responsive to checking means for enabling storage of data into the storage devices if the programs loaded into the system are free from virus.

The preferred embodiment of the present invention further comprises means for ensuring that programs to be loaded and executed in the computer system are free from virus, and means for removing viruses that may reside in the system.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating a computer system wherein the present invention is embodied;

Figure 2 is a flow diagram illustrating the typical power on operation of a computer system; and

Figure 3 is a block diagram of the preferred embodiment of the virus prevention system of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 is a block diagram of a typical computer system 100 including the preferred embodiment of the present invention. The computer system 100 comprises a CPU 10 which interacts through bus 11 with its other components, including a Random Access Memory (RAM) 12, a Read Only Memory (ROM) 13 and a plurality of

input/output (I/O) devices. These I/O devices 14 include basic devices such as data storage devices (e.g., floppy disks or hard disks), keyboard (not shown) and monitors (not shown). The I/O devices 14 may also include one or more called installable (optional) devices such as a tape or a CD-ROM device (not shown).

One of the storage devices 14 may be used to load a boot program at initialization (such as at power on). When so used, such storage device 14 is also called a booting device.

The ROM 13 contains software instructions commonly known as the Basic Input/Output System (BIOS) for performing interface operations with the I/O devices 14 of the computer system 100. Also stored in the ROM 13 is a software routine which operates to load a boot program from the booting device. The boot program will typically be executed when the computer system 100 is powered on or when initialization of the system is needed.

As typical with most of today's computer system, the I/O devices 14 communicate with the CPU 10 by generating interrupts. The CPU 10 distinguishes interrupts from among the I/O devices 14 through individual interrupt codes assigned thereto. Responses of the CPU 10 to I/O device interrupts differ, depending, among other things, on the devices 14 generating the interrupts. Interrupt vectors are provided to direct the CPU 10 to different interrupt handling routines. For example, in many of today's personal computers, two hundred and fifty six such interrupt vectors are provided.

The interrupt vectors are generated during initialization (or Boot) of the computer system 100 by execution of the BIOS. Because responses of the CPU 10 to device interrupts may need to be changed from time to time, the interrupt vectors may need to be modified from time to time so as to be able to direct the CPU 10 to different interrupt handling routines. To allow modification of the interrupt vectors, they are stored in the RAM 12 during operation of the computer system 100.

To increase efficiency and user friendliness, most computer systems typically are controlled by an operating system program (e.g., Microsoft's Disk Operating System, MSDOS). An operating system comprises modules for driving the basic I/O devices (such as the IO.SYS) and controlling other system control modules (e.g., MSDOS). In addition, the operating system may include a command interpreter (such as COMMAND.COM in MS DOS) for interpreting commands entered by a user.

Referring now to Figure 2, the prior art initialization process is illustrated when power of the computer system 100 is turned on (block 20), CPU 10 typically first performs a power-on self test ("POST") (block 21) stored as part of the BIOS in

the ROM 13. If there is no error in the POST, a boot program will be loaded into the computer system 100 (block 22) from the booting device. When the boot program executes without error, the operating system module (IO.SYS) will be loaded (block 23). This operating system module is a software driver for the basic I/O devices 14. The basic I/O devices 14 are then tested (block 24). Next, the core of the operating system (MSDOS.SYS) (block 25), the software drivers for the installable devices (block 26), and the command interpreter (COMMAND.COM) (block 27) are loaded and executed in succession.

After completion of the above-mentioned procedure, the booting operation is considered finished and the computer system is ready for normal operation. It can then receive commands from a user, whose commands may include the loading and execution of one or more application programs.

Computer viruses commonly in existence may be classified into: (1) Boot Time Infection Virus, (2) Program Infection Virus and (3) System Infection Virus.

Boot Time Infection Viruses typically affect a computer system by replacing the boot program with a virus program at boot time to control the system. To explain how this virus steals control of the computer system, the normal booting procedure is explained first.

If the computer system 100 is booted from a booting device that is infected with a virus (such as when an infected floppy diskette is inserted into a floppy disk drive), the virus alters one or more interrupt vectors (for example, the disk storage interrupt vector) during the booting procedure so that they will point to a virus program. The computer system 100 perceives the virus program as a portion of the BIOS. Thus, the virus program can steal control of the interrupt handling process for disk access. When the user accesses data from the disk, for example, the virus program can copy itself into the disk. Though the computer appears to operate normally, the disk in the computer system has already been infected.

Program Infection Viruses typically reside in application programs. They typically attack the computer system during execution of the application program by modifying the interrupt vectors so as to cause the system to execute a virus program.

System Infection Viruses typically reside in the operating system program (such as the IO.SYS, MS DOS.SYS). They attack the system during the booting process.

Figure 3 is a flow chart of a system initialization process similar to that shown in Figure 2. However, the process is improved to incorporate the present invention.

First, the system power is turned on and the CPU starts in step 20. At system power-on, if the POST (block 21) executes without error, the booting program would normally be loaded into the computer system 100 from the booting devices (block 22).

In accordance with the present invention, before the booting program is loaded and executed (block 22), a set/release switch 16 is set to inhibit writing to the storage devices 14 (block 21A). In other words, the setting of the set/release switch 16 would allow only retrieval of data from the storage devices (read-only) 14.

Methods for inhibiting storage of data into a particular storage device can be performed in several different ways. For example, since the bus controller 15 of the computer system 100 screens all I/O operations passing through the bus 11, it can, responsive to the setting of the set/release switch 16, reject write operations to designated I/O devices 14.

Alternatively, the set/release switch 16' may reside in the BIOS stored in ROM 13. Since all I/O operations are performed by execution of the BIOS, the BIOS can be implemented so that setting of the set/release switch 16' effectively causes the BIOS to ignore write instructions to certain devices.

Advantageously, the setting of the set/release switch 16 should only inhibit writes to non-volatile storage devices 14 (as compared to volatile storage devices), since integrity of content these devices need to be preserved.

By setting the set/release switch 16, a virus cannot attack the data/programs in the storage devices 14 at boot time.

Although the flow chart shows the setting of the set/release switch 16 right before booting of the computer system 100, it will be understood by those skilled in the art that the switch 16 can be set (to inhibit write operations) at any point in time before the system boots. By way of example, the set/release may be set by the system power-on signal.

After the set/release switch 16 is set, the computer system 100 can begin loading and executing the boot program and software drivers for the basic devices (blocks 22 and 23).

After these programs are loaded and executed, the computer system 100 will check to see if it has been attacked by a virus (Block 23A). This is done by comparing the interrupt vectors already loaded into RAM 12 against their corresponding normal (uninfected) values. The normal (uninfected) values of the interrupt vectors can be stored in any one of the storage devices 14.

Alternatively, since default values of the interrupt vectors typically point to the address range of the BIOS, the comparison may only check whether the value of a vector falls within such an appropriate range.

It will be understood by those skilled in the art that, depending on a cost/effectiveness tradeoff, only one or a particular set of interrupt vectors needs to be checked. For example, since most boot time infection viruses use the interrupt vectors of the monitor, it may be sufficient for the purpose of this invention to only check whether such interrupt vectors have been altered.

If the value of an interrupt vector has been changed, the computer system 100 will take appropriate corrective actions, such as producing a signal to notify the user that the booting program has been infected with a virus (block 2). In accordance with the preferred embodiment of the present invention, a virus indicator 17 is provided in one of the non-volatile storing devices. When the system discovers a virus in the booting program, the virus indicator 17 will be set to inform the user that the booting device is infected. The virus indicator 17 is also stored in a non-volatile storage device.

If the interrupt vectors have not been contaminated, power on operation of the computer system will continue with the loading of the system driver (MSDOS.SYS) (block 25) and the installable device driver (block 26). In accordance with the preferred embodiment, however, each such program will be loaded only after it is determined that no virus resides therein (blocks 24A and 25A).

As an alternative implementation, a copy of the booting program in its uninfected state, can be stored in the computer system 100, either in a ROM 13 or in the non-volatile storage devices 14. This copy can be used to compare the boot program from the booting device before the booting.

According to the preferred embodiment, although several program modules (MSDOS.SYS, installable device drivers and COMMAND.COM) still need to be loaded, but since means are provided to check the integrity of these programs before they are loaded, the set/release switch 16 or 16' can be reset at this time (step 24) if the interrupt vectors in the RAM 12 are not altered. If such means are not provided, the set/release switch 16 or 16' may be reset near the end of the power on process in step 26C.

In the preferred embodiment, a check sum for each of the system modules (MSDOS.SYS and installable device drivers) in their uninfected states is stored. Before each of these modules is loaded and executed, the corresponding check sum is calculated and compared with stored values to determine whether any of the operating system modules are infected. If the comparison shows no alterations, the system will load the programs and proceed normally. Otherwise, a warning signal and the virus indicator 17 are set to inform the user to

replace the booting device.

The power-on procedure is now almost completed and the CPU 10 is ready to work. However, to protect the system from file-type (or program-type) viruses, a verification program is provided in the preferred environment for preventing program-type virus from entering into the computer system. (Blocks 26A and 26B). The verification program requires a user give the system a specific characteristic (e.g., check sum) of each application program in its uninfected state.

The system will check the characteristic of an application program against the characteristic given in step 26B. Only if such characteristic matches will the program be allowed to execute.

In accordance with another preferred embodiment of the present invention, a virus removing means is also provided. The virus clearing program will remove viruses detected at the above steps. The virus removing means stores a good copy of the booting program, the operating system modules and the installable device drivers. If a virus was detected in a previous booting process, the good copy of the program will be copied from the nonvolatile read-only storage means over the bad copy that was used in the previous booting process. It should be understood by those skilled in the art that one of the above described virus detection methods may also be used to detect the presence of viruses in the verification program and the virus removing program virus.

The protection system of the present invention eliminates drawbacks of virus prevention during the booting procedure. The protection system controls the booting procedure to exactly and efficiently prevent the system from being contaminated with computer virus. Furthermore, the preferred embodiment of the prevention system prevents damage from program-type virus and provides a virus-clear means. Such efficiency is unattainable by an conventional prevention system. Thus, the prevention system of the present invention provides comprehensive protection from damage caused by computer viruses.

Of course, it is understood that the above is merely a preferred embodiment of the invention and that various changes and alterations can be made without departing from the true spirit and broader aspects thereof.

Claims

1. A computer system having protection against vandalism by a virus, the system comprising:
   at least one storage device capable of inputting a booting program into the system and storing data; and
   means for preventing the virus from affect-

ing data in the storage device at boot time.

2. The system of claim 1 or 2, wherein the preventing means comprises means for inhibiting storage of data before leading of the boot program.

3. The system of claim 1, wherein the preventing means comprises means for detecting presence of a virus at boot time.

4. The system of claim 3, wherein the system interacts with a plurality of input/output devices by responding to a set of interrupt vectors, and said detection means comprises means for checking whether said interrupt vectors are modified.

5. The system of one of claims 1 to 4, further comprising means for storing predefined characteristics of programs and means for comparing said stored characteristics against characteristics of corresponding programs to be loaded into the system.

6. A method for protecting a computer system against vandalism by a virus, the system having at least one storage device capable of inputting a booting program into the system and storing data, the method comprising the steps of:
   booting the system, and
   preventing the virus from affecting data in the storage device during the booting.

7. The method of claim 6, wherein the preventing step comprises the sub step of inhibiting the storage of data before loading the boot program.

8. The method of claim 6 or 7, wherein the preventing step comprises the step of detecting the present of a virus at boot time.

9. The method of claim 8, wherein the system interacts with a plurality of input/output devices by responding to a set of interrupt vectors and said detection step comprises the step of checking whether said interrupt vectors are modified.

10. The method of one of claims 6 to 9, further comprising the steps of:
   storing predefined characteristics of programs; and
   comparing the stored characteristics of programs against the characteristics of corresponding programs being loaded into the

system.

5

10

15

20

25

30

35

40

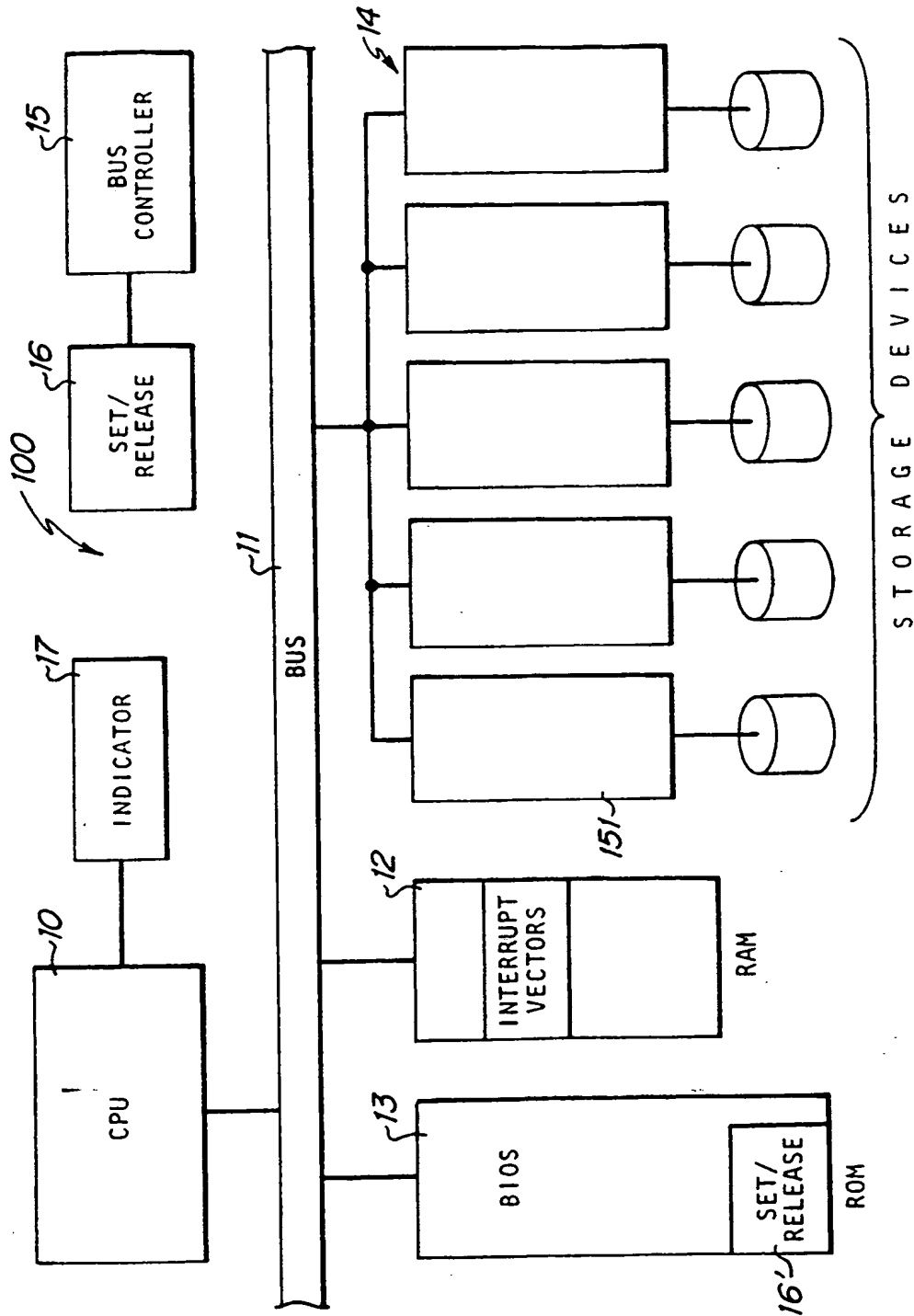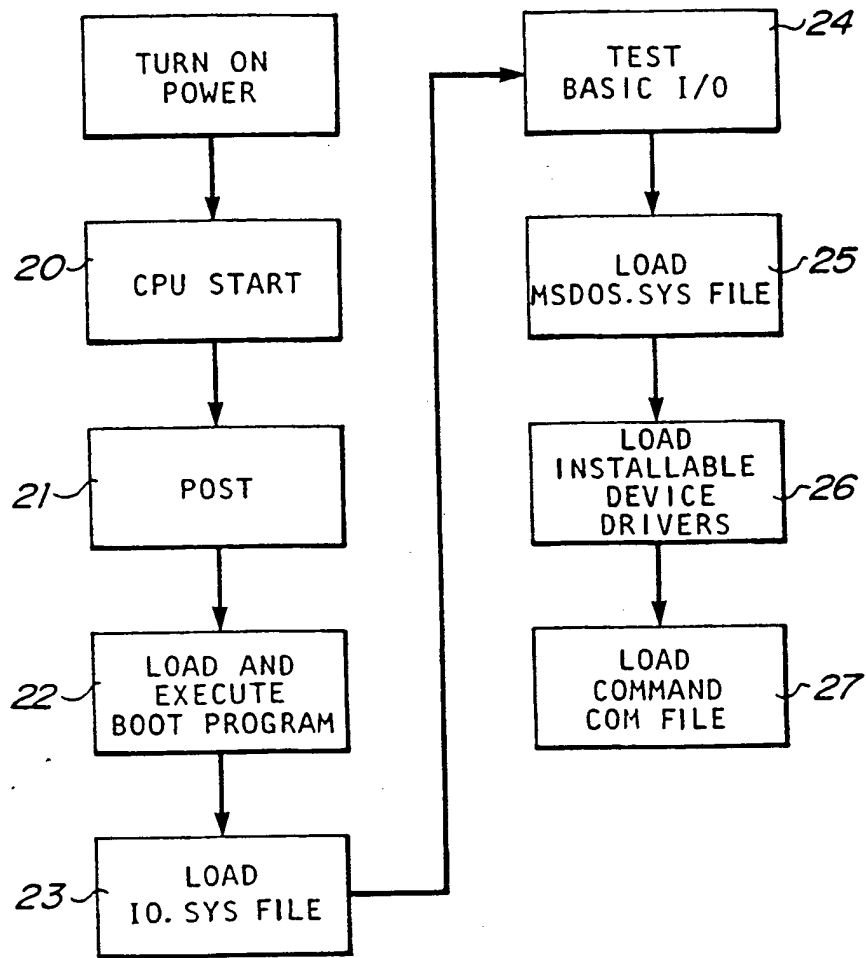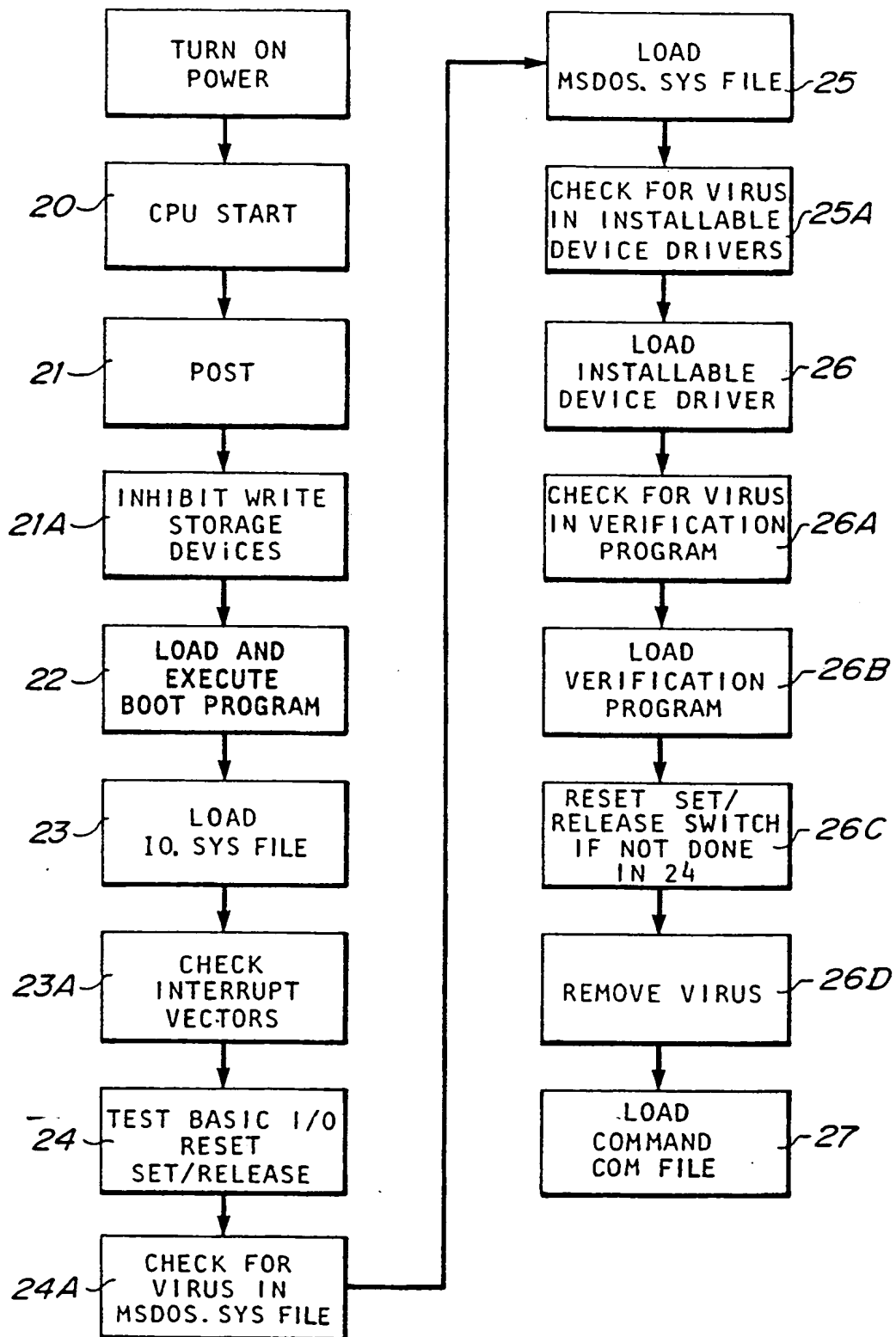45

50

55

Figure 1

Figure 2   (Prior Art)

```
    ┌──────────────┐                      ┌──────────────┐
    │  TURN ON     │                      │    LOAD      │
    │  POWER       │              ┌──────▶│ MSDOS. SYS FILE│─25
    └──────────────┘              │       └──────────────┘
           │                      │              │
           ▼                      │              ▼
    ┌──────────────┐              │       ┌──────────────┐
20──│  CPU START   │              │       │ CHECK FOR VIRUS│
    └──────────────┘              │       │ IN INSTALLABLE │─25A
           │                      │       │ DEVICE DRIVERS │
           ▼                      │       └──────────────┘
    ┌──────────────┐              │              │
21──│    POST      │              │              ▼
    └──────────────┘              │       ┌──────────────┐
           │                      │       │    LOAD      │
           ▼                      │       │ INSTALLABLE  │─26
    ┌──────────────┐              │       │ DEVICE DRIVER│
    │ INHIBIT WRITE│              │       └──────────────┘
21A─│  STORAGE     │              │              │
    │  DEVICES     │              │              ▼
    └──────────────┘              │       ┌──────────────┐
           │                      │       │ CHECK FOR VIRUS│
           ▼                      │       │ IN VERIFICATION│─26A
    ┌──────────────┐              │       │ PROGRAM      │
    │ LOAD AND     │              │       └──────────────┘
22──│ EXECUTE      │              │              │
    │ BOOT PROGRAM │              │              ▼
    └──────────────┘              │       ┌──────────────┐
           │                      │       │    LOAD      │
           ▼                      │       │ VERIFICATION │─26B
    ┌──────────────┐              │       │ PROGRAM      │
    │    LOAD      │              │       └──────────────┘
23──│ IO. SYS FILE │              │              │
    └──────────────┘              │              ▼
           │                      │       ┌──────────────┐
           ▼                      │       │ RESET SET/   │
    ┌──────────────┐              │       │ RELEASE SWITCH│─26C
    │  CHECK       │              │       │ IF NOT DONE  │
23A─│  INTERRUPT   │              │       │   IN 24      │
    │  VECTORS     │              │       └──────────────┘
    └──────────────┘              │              │
           │                      │              ▼
           ▼                      │       ┌──────────────┐
    ┌──────────────┐              │       │ REMOVE VIRUS │─26D
    │ TEST BASIC I/O│             │       └──────────────┘
24──│ RESET        │              │              │
    │ SET/RELEASE  │              │              ▼
    └──────────────┘              │       ┌──────────────┐
           │                      │       │    LOAD      │
           ▼                      │       │ COMMAND      │─27
    ┌──────────────┐              │       │ COM FILE     │
    │ CHECK FOR    │              │       └──────────────┘
24A─│ VIRUS IN     │──────────────┘
    │ MSDOS. SYS FILE│
    └──────────────┘
```

Figure 3

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	WO-A-9 005 418 (S. LENTZ) * the whole document *	1-3,6-8	G06F1/00 G06F11/00
E	WO-A-9 113 403 (RODIME PLC) * the whole document *	1-3,6-8	
A	DE-A-3 736 760 (TRANS TECH TEAM IMMOBILIEN GMBH) * the whole document *	1-3,5-8,10	
A	CHIP no. 6, June 1989, WURZBURG DE pages 72 - 75; D. TREPLIN: 'Virenschutzprogramme' * the whole document *	1,4-6,9,10	
A	FR-A-2 629 231 (J.-L. SALZMANN) * the whole document *	1,2,6,7	
A	GB-A-2 231 418 (S & S ENTERPRISES (AMERSHAM) LTD.) * the whole document *	1,2,6,7	TECHNICAL FIELDS SEARCHED (Int. Cl.5)
A	GB-A-2 222 899 (ANTHONY MORRIS ROSE) * the whole document *	1,2,6,7	G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	20 FEBRUARY 1992	DURAND J.